

Please amend the Specification as follows:

(page 1, lines 2 and 3 of the original specification)

RELATED APPLICATIONS:

B¹ This application is a ~~continuation of provisional application serial number 60/164,437~~
~~filed November 9, 1999.~~ continuation-in-part of application serial number: 09/120,636, filed
July 22, 1998—now issued patent number: 6,272,131, issued August 7, 2001.

(page 18, lines 19-21 of the original specification)

B² FIG. 24A is a functional diagram of a switch with the ~~FAST~~ Fast Switching mode of
operation, which implies that there are pre-computed schedules for transferring the incoming
data packets to their respective output ports;

(page 19, lines 4-7 of the original specification)

B³ FIG. 25 provides an example of a fabric controller that uses a plurality of ~~FAST~~ Fast
switching matrices, where there is a different switching matrix for a subset of time slots in every
time frame, for each time frame in every time cycle, and for each time cycle in every super-cycle
in accordance with the present invention;

(page 20, lines 14-16 of the original specification)

B⁴ FIG. 32B is a timing diagram of a switching operation that is responsive to the common
time reference 002 with three pipeline forwarding phases that enable the operation with the pre-
computed schedules with the ~~FAST~~ Fast Queuing Method;

(page 22, lines 3 and 4 of the original specification)

B⁵ FIG. 45 is a functional block diagram of an SVP (synchronous virtual pipe) interface with
per time frame queues;

(page 22, lines 14-17 of the original specification)

B⁶ FIG. ~~49~~ is a diagram of an ~~8 by 8 multi-stage interconnection switch that is constructed~~
~~of 2 by 2 switching elements;~~

FIG. 49A is a straight connection of a 2-by-2 switching block;

CONT
B6
FIG. 49B is a cross connection of a 2-by-2 switching block;

FIG. 49C is a diagram of an 8-by-8 multi-stage interconnection switch that is constructed of 2-by-2 switching elements;

FIG. 50A is a comparison table of a multi-stage interconnection switch with a crossbar switch; and

(page 45, lines 15-20; page 46, lines 1 and 2 of the original specification)

7
B7
The fifth operation of 35-04 saves the routing information in the ROUTE-STORE variable information that will be used to skip the scheduling step for the successive data packet with the same PID. These packets will be routed into the ~~FAST~~ Fast part of the queues B-1 through B-k' in FIGS. 9 and 10.

In step 35-06 in FIG. 8 for L1/L2=01 or L1/L2=10 a data packet is stored in the ~~FAST~~ Fast part of the queues B-1 through B-k' in FIGS. 9 and 10, and consequently this data packet receives the same schedule to be transferred across the switch as previous data packets with same PID.

?
(page 47, lines 7-12 of the original specification)

B8
Within each of the queues B-1, B-2, ² ... and B-k' are a plurality of sub-queues CBR, VBR, ~~FAST~~ Fast, and MCST (Multicast). (This is not shown explicitly, since multicast implies that a data packet is copied to multiple queues to multiple output ports.) The sub-queues are used to differentiate between the different types of data packet traffic entering each queue, as constant bit rate (CBR), variable bit rate (VBR), best-effort, and ~~FAST~~ Fast (for data with pre-computed switching schedules).

(page 48, lines 9-14 of the original specification)

B9
The SBCC 36D is constructed of a central processing unit (CPU), a random access memory (RAM) for storing data packets, and a read only memory (ROM) for storing the select buffer and congestion controller processing program. The SBCC is additionally coupled to the RAM 36C by read signals 36R1, 36R2, and so forth, respectively to queues B-1, B-2, and so forth. The signals 36R1, 36R2 et. al., permit the SBCC to select which of the sub-queues (e.g., CBR, VBR, ~~FAST~~ Fast) of queues B-1, B-2 et. al., to read.

(page 63, lines 1-13 of the original specification)

Method 1: ~~FAST~~ Fast switching (following FIGS. 24-25)

10
11
In ~~FAST~~ Fast switching an incoming data packet is switched, by the routing controller 35B in FIG. 7, to the one or more queues, selected from 36-1 through 36-N, that are associated with the output ports the incoming data packet should be forwarded from. The data packet is stored by the packet scheduling and rescheduling controller (PSRC) in the ~~FAST~~ Fast part of one of the B-1 through B-k' in FIG. 9.

Data packets that are stored in the ~~FAST~~ Fast part of a queue have pre-computed schedules for being switched from input to output, and therefore, skip phase 2 of scheduling and rescheduling at $TF(t+1)$, as shown in FIG. 15. Instead as illustrated in FIG. 24, there are only three pipelined forwarding phases for forwarding data packets as in the present invention. The phases are numbered phase 1', phase 2', and phase 3'. In the preferred embodiment, each phase is accomplished over a period of time equal to one time frame.

(page 64, lines 12-21; page 65, lines 1-21 of the original specification)

11
12
The fast switching from the ~~FAST~~ Fast queues is performed in accordance to switching information stored in a plurality of switching matrices 2500 in FIG. 25. In general, there is a different matrix for every time slot. Therefore, if there are s - slot positions in a time frame, f frame positions in a time cycle, and c cycle positions in a super-cycle, then the total number of switching matrices 2500 $S(i,j,t)$, is $s*f*c$. In $S(i,j,t)$ the variable i indicates the time slot position in the time frame, the variable j indicates the time frame position in the time cycle, the variable t indicates the time cycle position in the super-cycle.

Each switching matrix has an element for each input-output pair, consequently, if there are four input ports and four output ports the total number of elements in each matrix is sixteen, as shown, for example, in FIG. 25. The value in the elements in each matrix can be of two types: type =0 - temporary value in this switching matrix, and therefore, used only once, and type =1 - permanent value in this switching matrix, and therefore, used multiple times.

For switching out of the ~~FAST~~ Fast queue, the permanent values are used. If the traffic pattern is fixed the switching matrices contain only permanent values.

In Method 2 below, it is shown how setting up the permanent values in the switching

matrices can be done on the fly by the next data packet in the stream.

Method 2: "Train" switching through the ~~FAST~~ Fast queues

The objective of "train" switching is twofold:

1. To avoid the Phase 2 (the scheduling and rescheduling operations) in FIG. 15 – as much as possible, and
2. To avoid the need of setting up the permanent values in the switching matrices prior to the transmission of data packets of a real time flow.

CONT
B11
There are various ways to achieve the above two objectives. One simple way is using the first set data packets in the time frame, time cycle or super-cycle for setting up the permanent values in the switching matrices **2500** in FIG. 25. For example, if a certain PID has a transmission pattern of three data packets that are transmitted in three predefined time frames of each time cycle, then the first three data packet will use Phase 2 (the scheduling and rescheduling operations) in FIG. 15 – while subsequent data packets over this PID will be switched from the ~~FAST~~ Fast queues using the permanent values as specified in Phase 2' in FIG. 25.

(page 66, lines 16-20 of the original specification)

12
B
Note, as shown in FIGS. 9 and 10, per time frame queuing is performed, that every phase in FIGS. 15 and 24 is one time frame, and that the order of transmission of different flows from the same ~~FAST~~ Fast queue can be arbitrary. This fact simplifies the scheduling and timing requirement from the switch design and distinguishes this approach from circuit switching.

(page 73, lines 18-20 of the original specification)

13
B
FIG. 32B shows a three phase operation of the method that is based on the ~~FAST~~ Fast Queues (as were shown in FIGS. 9 and 10) in which there are pre-computed switching schedules for the incoming data packets.